Implémentations de plates-formes dynamiques de services sur .Net

Clément ESCOFFIER
Didier DONSEZ

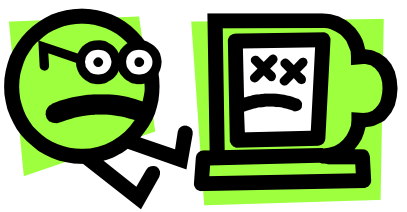
Laboratoire LSR – Equipe ADELE

Informatique et Mathématiques Appliquées de Grenoble

Motivations

- Applications dynamiques
- Architectures orientées services dynamiques centralisées
- OSGi (OSCAR)

- Migrer OSCAR sur .NET

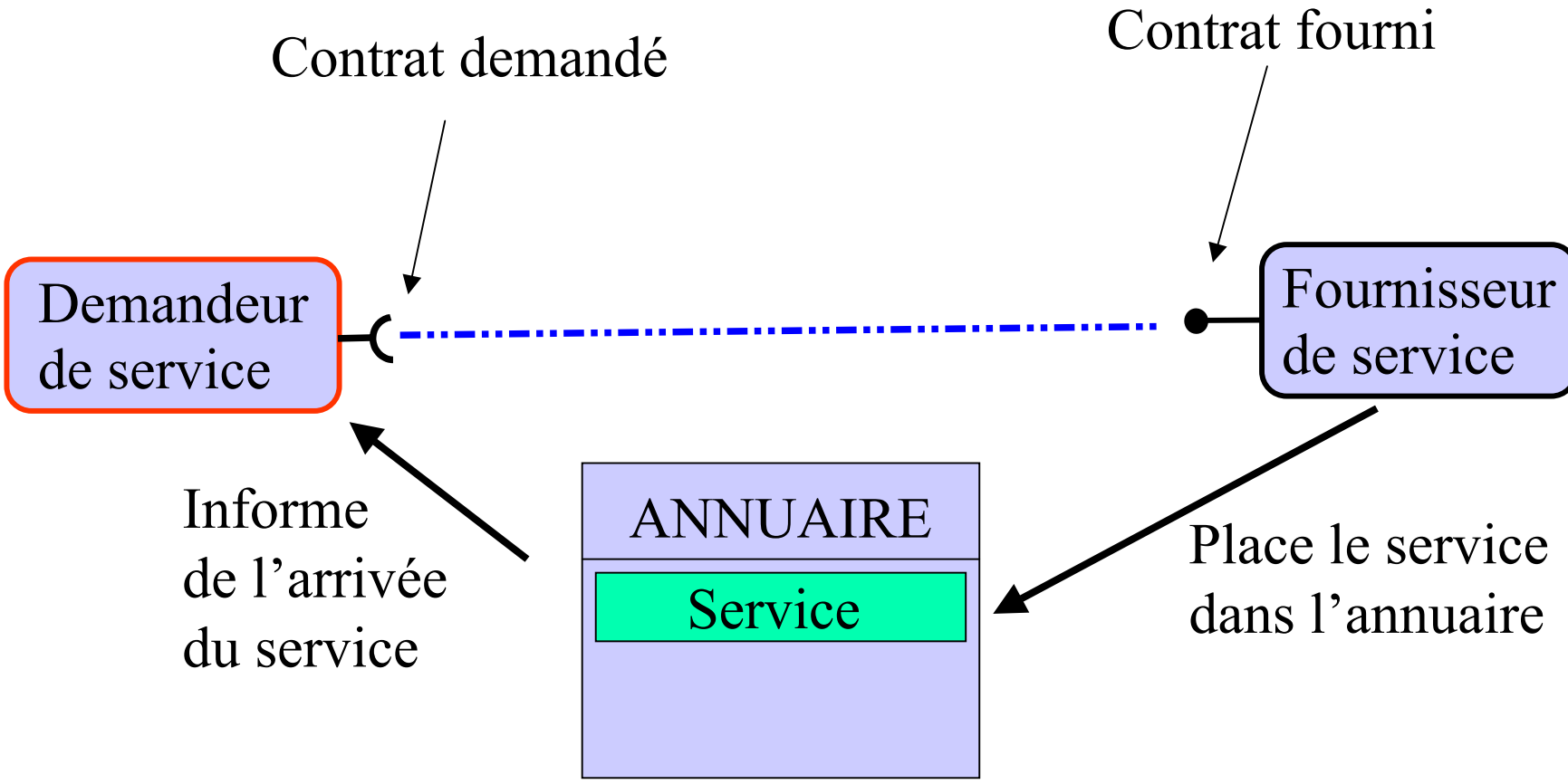


- Proposer des alternatives proches d'OSGi pour .NET

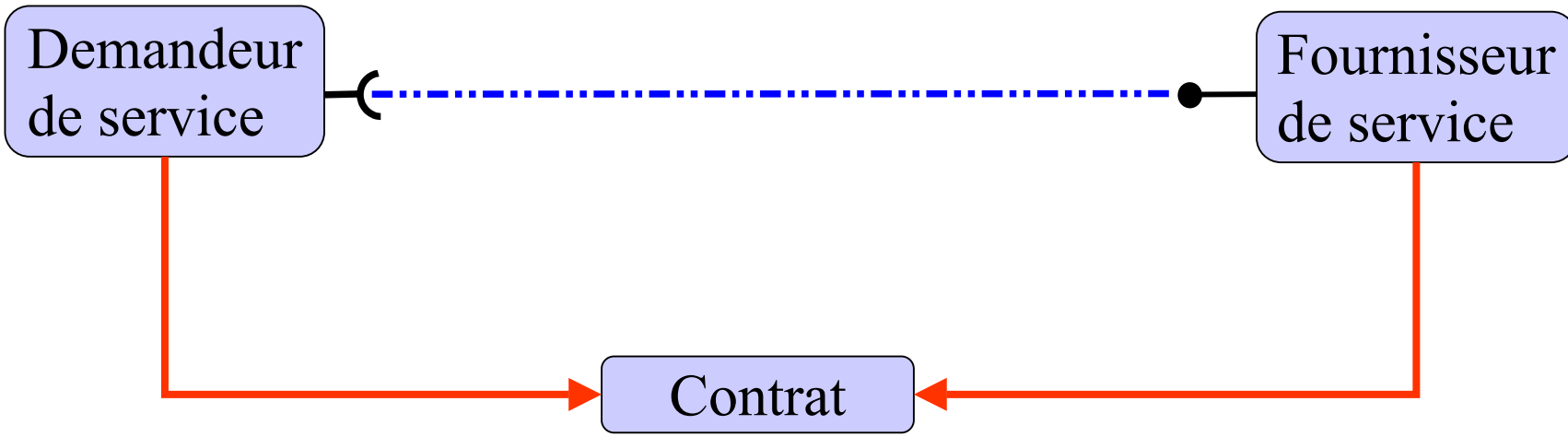
Plan

- Architectures Orientées Services Dynamiques
- Plates-formes de Services dynamiques
- La plate-forme .NET
- Les 4 Alternatives développées
- Récapitulatif
- Perspectives & Conclusion

rchitectures Orientées Services Dynamique



chitectures Orientées Services Dynamique
Chargement de classes



- Classe partagée
- Classe chargée 2 fois
- Souche / Squelette

Architectures Orientée Services Dynamique
Technologies Actuelles

	Invocation	Annonce de départ	Type d'annuaire	Langage
JINI	Distante (RMI)	Bail	Réparti (ad-hoc)	JAVA
OpenWings	Distante (RMI IIOP)	Connecteur	Réparti (?)	JAVA
CORBA Trader	Distante (IIOP)	Non	Réparti (?)	Tous
UPnP	Distante (HTTP SOAP)	Non	Réparti (ad-hoc)	Tous
Web Services	Distante (HTTP SOAP)	Non	Centralisé (répliqué)	Tous
OSGi	Locale (Référence)	Évènement	Centralisé	JAVA

Plates-formes de Services Dynamiques

Permettre le déploiement d'applications orientées services de manière centralisée :

» Gestion du cycle de vie des services
» Gestion des liaisons entre services

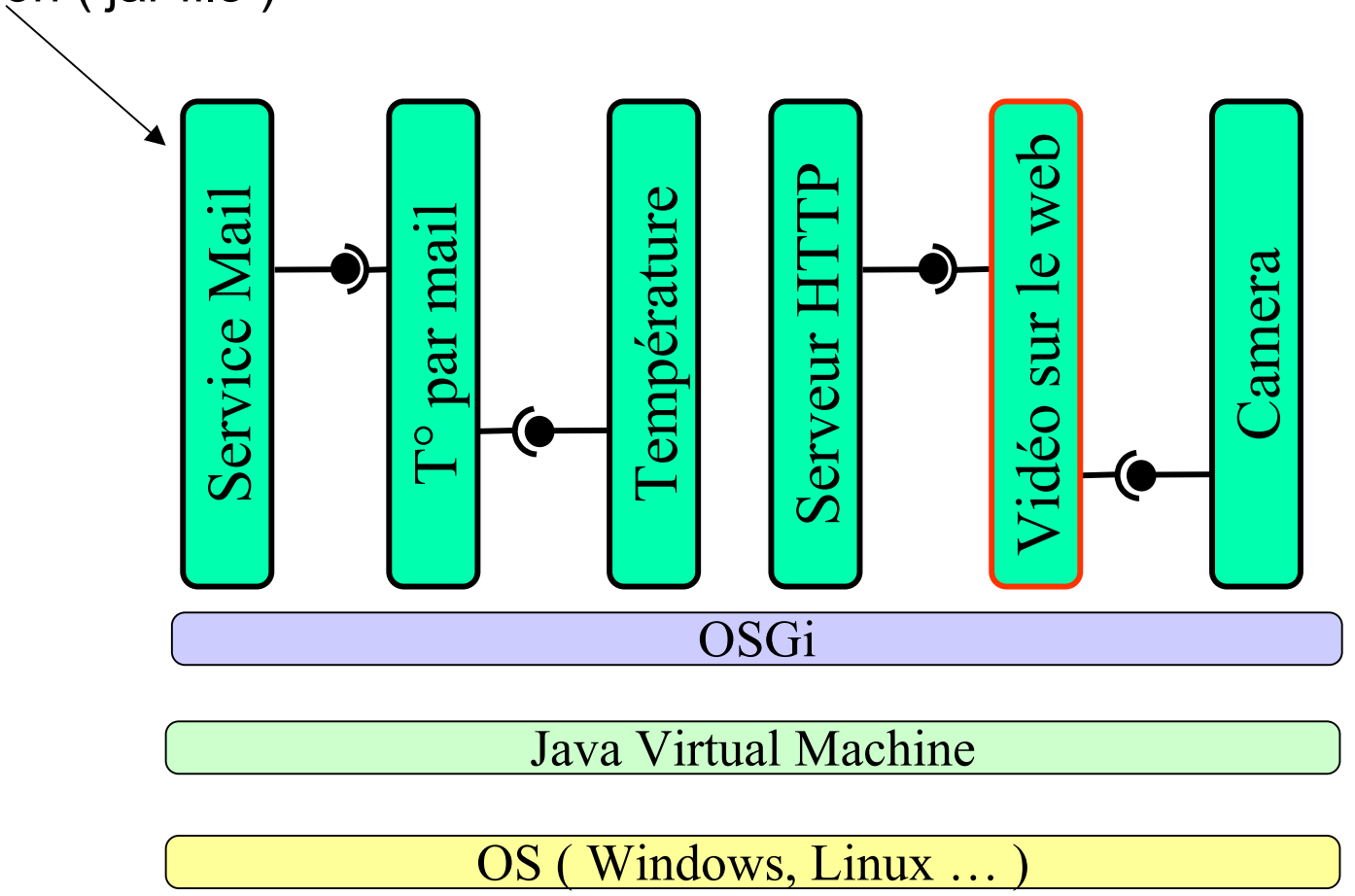
Sans interruption de la plate-forme

Caractéristiques :

- Prise en compte des arrivées et des départs des services
- Chargement dynamique de code
- Déchargement dynamique partiel de code
- Invocation de services
- Sans interruption de fonctionnement

OSGi : Architecture

undle :
Propose un ou plusieurs service
Requiert d'autres services
Unité de déploiement et
e livraison (jar file)

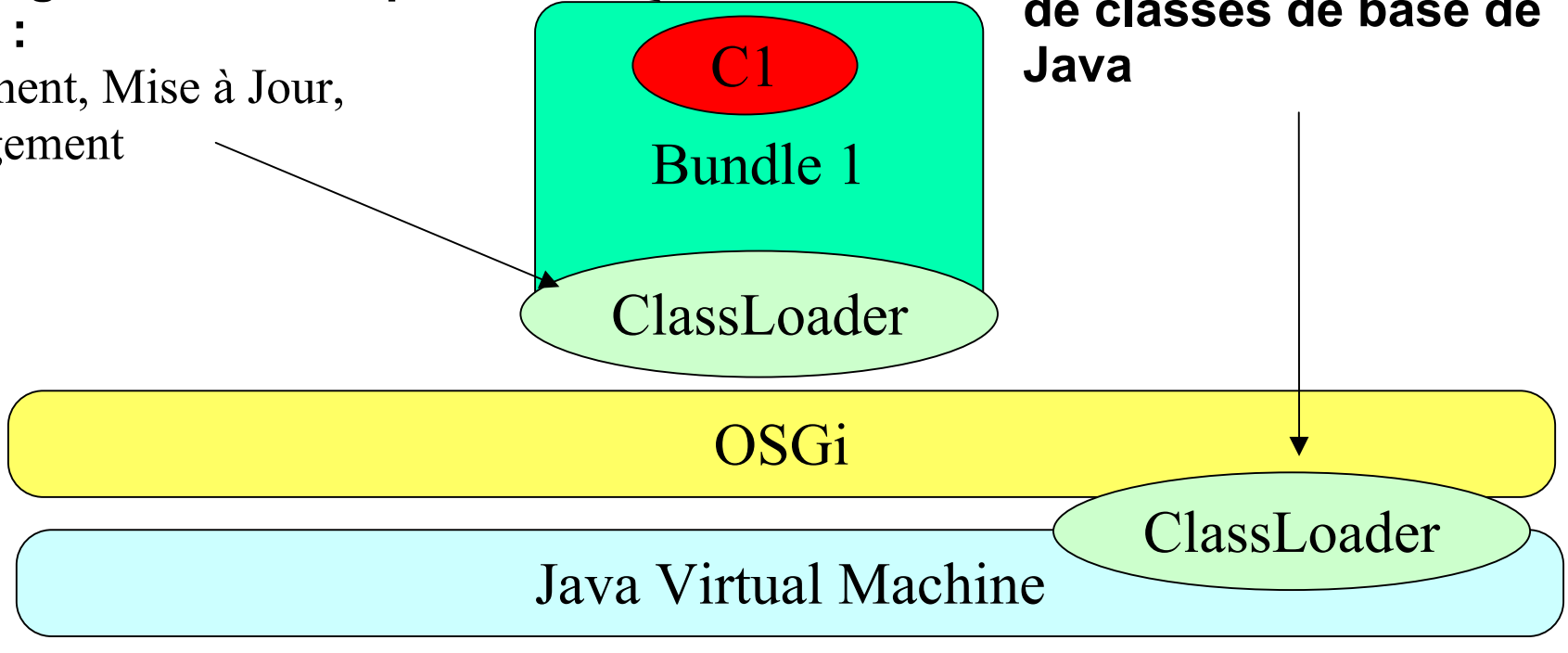


Chargement des classes dans OSGi : Anatomie d' un bundle

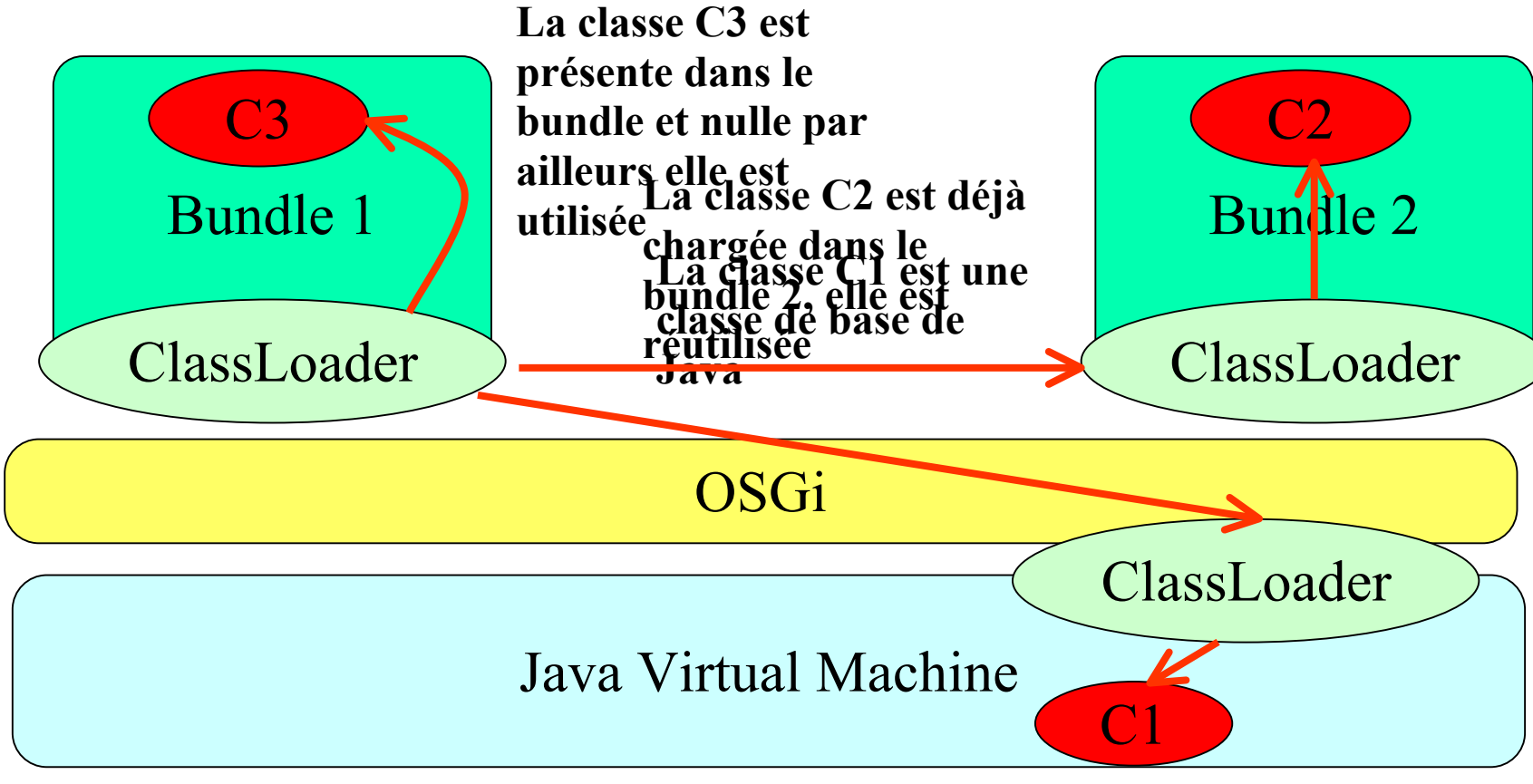
Classes présentes dans le bundle

n Chargeur de classe par ndle :
argement, Mise à Jour,
echargement

Système de chargemen de classes de base de Java



Chargement des classes dans OSGi :



.Net

NET en quelques mots :

- Une machine virtuelle nommée CLR *(JVM)*
- Interprète du MSIL *(ByteCode)*
- Supporte plusieurs langages : C#,J#... *(Java)*
- Unité de déploiement : Assembly *(Jar, Classe)*
- Environnement d'exécution et chargeur : domaine d'applic
- Possibilité de décharger du code (domaine d'application)

Plusieurs implémentations dont :

- La plate-forme standard de Microsoft
- ROTOR – Shared Source CLI (Microsoft)
- MONO (open source)
- Compact .NET Framework

Chargement des classes dans .Net

Un Resolveur d' assemblies par Domaine d' Application

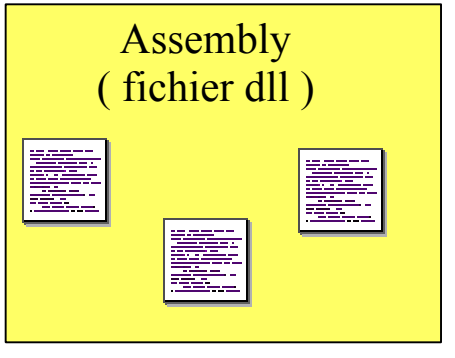
Permet le chargement d'assemblies

Le principe de chargement :

- Une classe d'une assembly déjà chargée est forcément réutilisée

- .Net charge des assembly et non pas les classes individuellement :

  - Recherche dans le Global Assembly Cache
  - Parcours d'emplacement physique
  - Microsoft Windows Installer

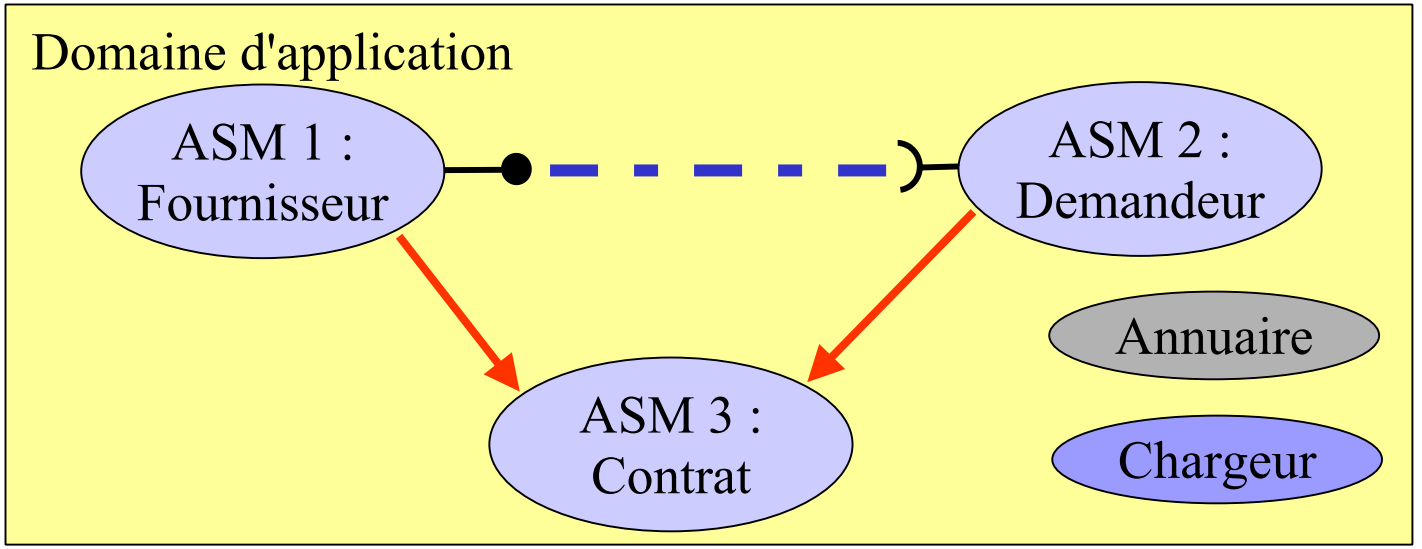


Alternatives de plates-formes de services sur .Net

- 4 alternatives développées
- Caractéristiques :
  - Prise en compte des arrivées et des départs des services
  - Chargement dynamique de code
  - Déchargement partiel de code
  - Invocation de services en direct ou via un proxy
  - Partage des attributs statiques
  - Masquage des classes d'implémentations
  - CLR standard ***VS*** CLR modifiée

Alternative 1 : Mono-domaine d'application

- Idée :
  - Un bundle ⇔ Une Assembly
  - Un seul domaine d'application



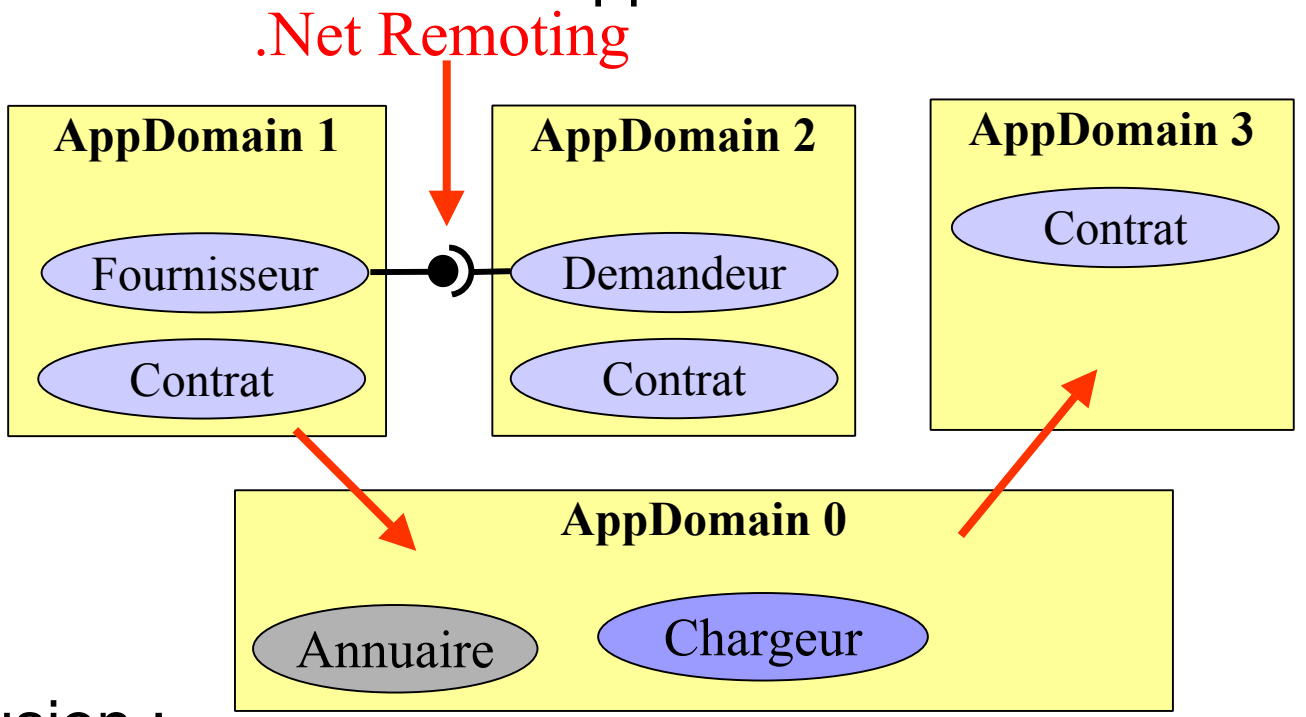
- Conclusion :

Déchargement partiel impossible

Visibilité des classes d'implémentations

Alternative 2 : Multi-domaines d'application

- Idée :
  - Un bundle ⇔ Un domaine d'application
  - Plusieurs domaines d'applications



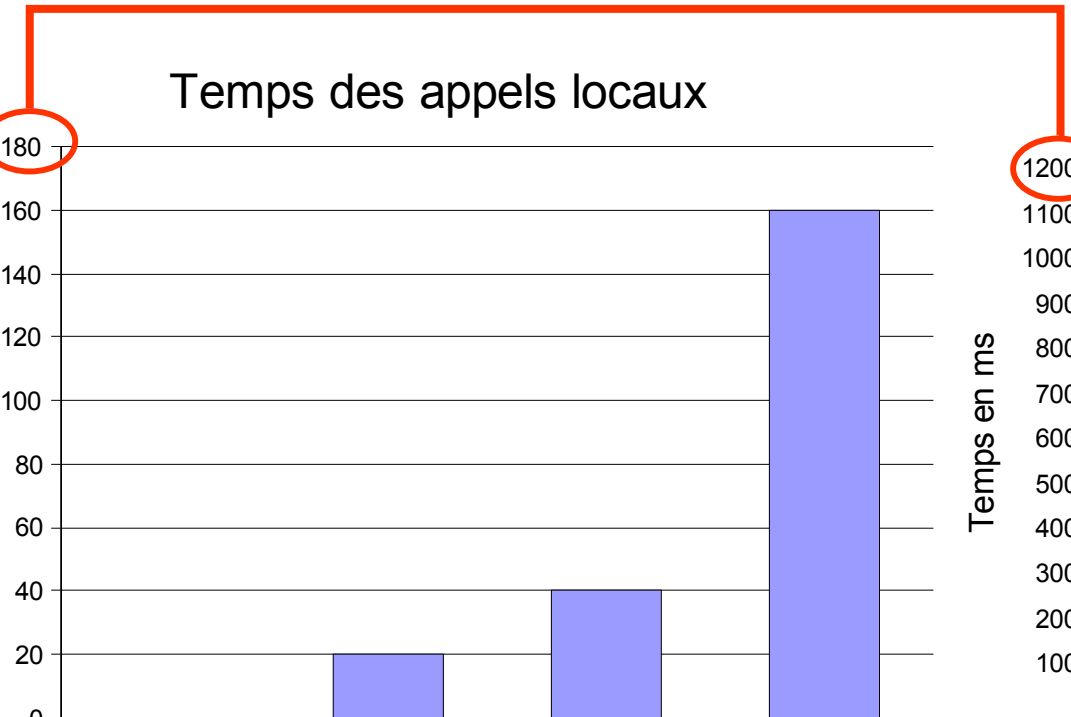
- Conclusion :

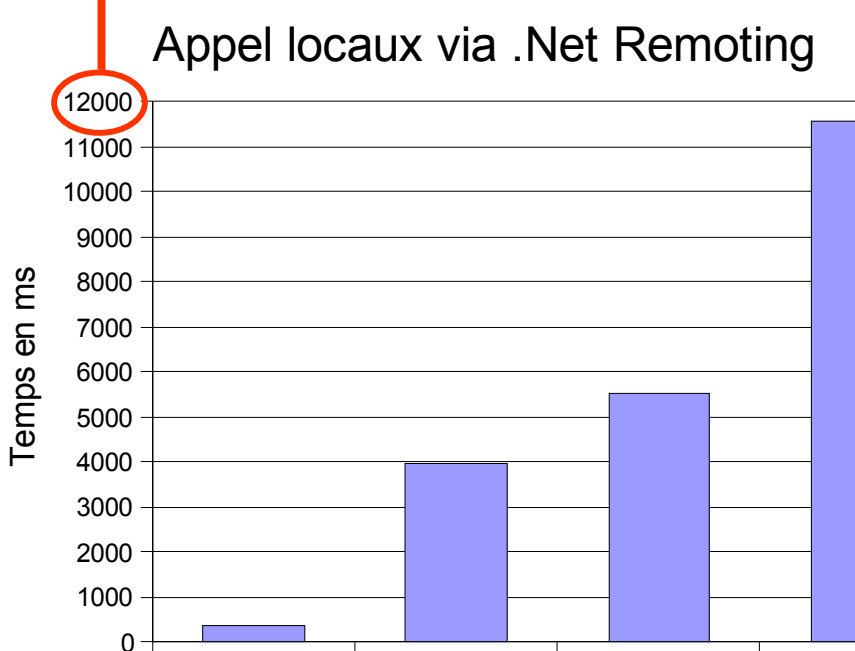
.Net Remoting apporte un surcoût lors de l'invocation

Mais prise en compte de plusieurs versions de contrats

Benchmark

- Benchmark mesurant les temps d'appel de méthode en ms
- Une seule CLR
- Appels de méthodes intra-domaine
- Appels de méthodes inter-domaine (.Net Remoting)
- 4 types d' objets échangés
- Machine: 1Ghz, 768 Mo de RAM, Windows XP SP2, CLR 1.1





Etude d'un implémentation de .Net : ROTOR

Shared Source CLI :

- Proposé par Microsoft
- Proche de la plate-forme .Net officielle
- Fournit un environnement .Net complet (compilateurs,VM ...)
- Destiné aux expérimentations et à la recherche

Quelques chiffres :

- 1,9 millions de lignes de code :
  - 1,1 millions en C++
  - 600 mille en C#
  - 125 mille en IL
  - Assembleur
- 5.900 fichiers sources / 9700 fichiers au total
- Noyau (mscorlib) : 867 classes en C++

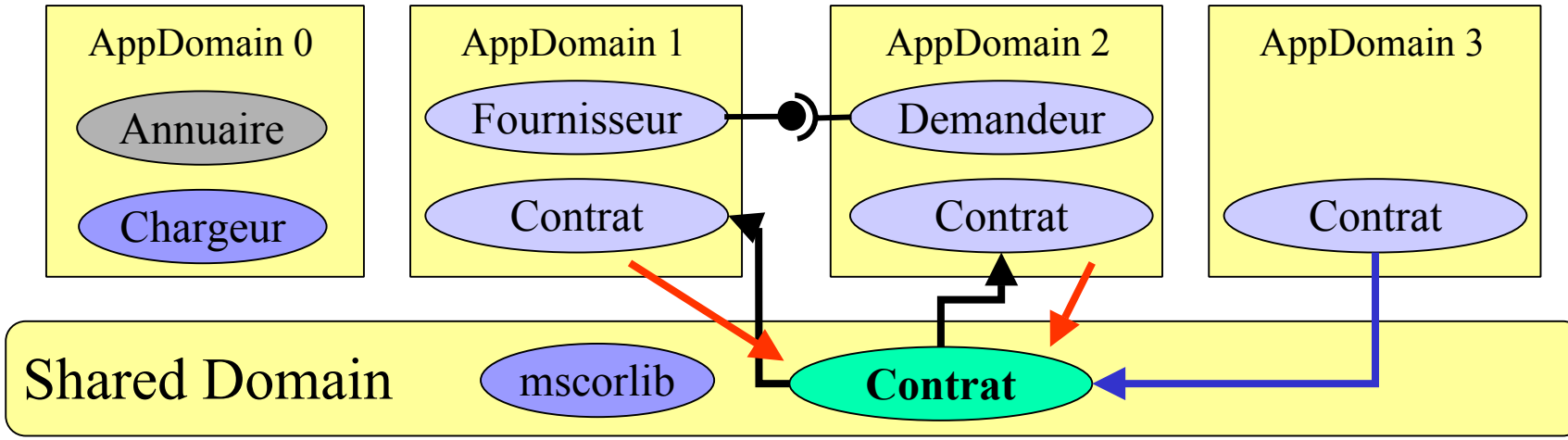
Ressources :

- News-groups de Microsoft sur le CLR et sur ROTOR
- Livre : Shared Source CLI (O'Reilly, mais écrit par Microsoft)

Utilisation du Shared Domain

- Idée : Utiliser le Shared Domain :
  - Possibilité de partager du code entre les domaines d'applications
  - Code exécuté dans les domaines d'applications

    mais compilé une seule fois pour tous



- Conclusion :


Les Contrats ne peuvent pas être mis à jour

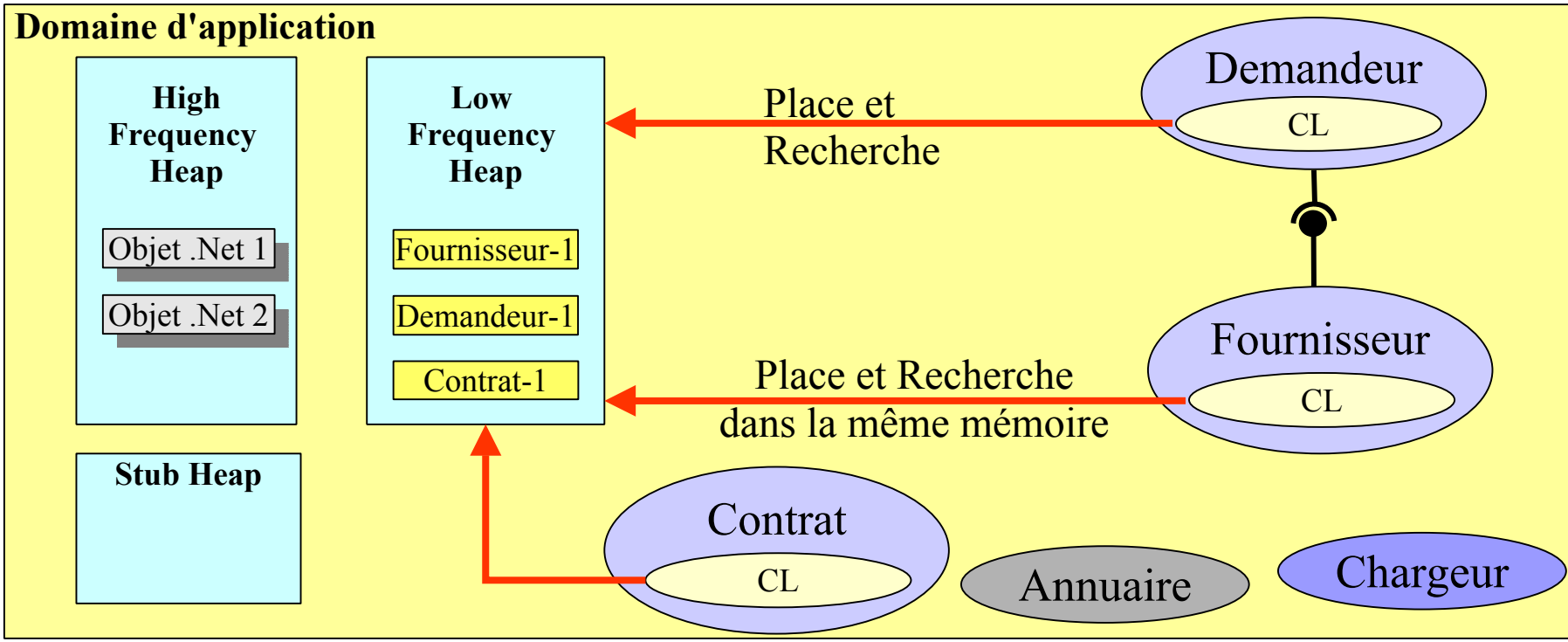

Prochainement caché


Invocation de services toujours via .Net Remoting

Alternative 4 : modification de la CLR

Chargeur de classe interne des assemblies :

- Chaque assembly possède son propre chargeur de classe
- Gère des objets EEClass dans le Low Frequency Heap



- Conclusion :

Inachevé : Corruption du Low Frequency Heap

Décourageant, modification de la CLR

Récapitulatif

	1	2	3	4	Référen
Aspect étudié	***Mono-domaine d'application***	***Multi-domaines d'application***	***Utilisation du Shared Domain***	***Modification de la CLR***	***OSGi***
Apparition dynamique des services					Oui
Chargement dynamique de code					Oui
Déchargement partiel de code					Oui
Type d'invocation					Direct
Partage des attributs statiques	Partagé	1 copie par domaine d'application	1 copie par domaine d'application	Partagé	Partag
Version de la machine virtuelle	Standard	Standard	Standard	ROTOR modifié (chargement de classe)	JVM dep version

OSGI.NET ?

A l'heure actuelle :

- Impossibilité de décharger les assemblies individuellem
- Personalisation du classloading interne difficile
- Visibilité des classes non personnalisable

Pas de manière acceptable

Perspectives :

- Déchargement d'assembly (version 4 ?)
- Transparence du Shared Domain

Mais ceci ne suffira pas pour implémenter OSGi.Net

Conclusion

Peut on implémenter une plate-forme dynamique de services sur .NET ?

Oui, en faisant des concessions :

- Suppression / Mise à jour de services
- Temps d'invocation
- Visibilité des classes d'implémentations
- Utilisation d'une CLR standard ou personnalisée

Des questions ?